

## Requirement:

## Create a Jenkins Job that will take two inputs:

- 1) Instance Id
- 2) Instance Type

## The Jenkins Job should perform 3 operations:

- 1) Change the instance type of instance with the provided instance type.
- 2) Perform an operation using ansible on the provided instance
- 3) Scale back instance to the previous instance type.

- Launch an ec2 instance for Jenkins\_server and testing\_server – for testing\_server I've associated ElasticIP

The screenshot displays the AWS Management Console interface. The top section shows a list of EC2 instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Two instances are listed: Jenkins\_Server (i-06a3f451908156cc0) and Testing\_Server (i-095c6b04cca499258), both running on t2.micro instances in us-east-1b. Below this, a filtered view shows only the Jenkins\_Server instance. The bottom section provides detailed information for the Jenkins\_Server instance, including its ID, public IPv4 address (44.204.38.31), and various tabs for configuration and monitoring.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Jenkins_Server	i-06a3f451908156cc0	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b
Testing_Server	i-095c6b04cca499258	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b

Name	Instance ID	Instance state	Instance type
Jenkins_Server	i-06a3f451908156cc0	Running	t2.micro
Testing_Server	i-095c6b04cca499258	Running	t2.micro

### Instance: i-06a3f451908156cc0 (Jenkins\_Server)

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
<b>Instance summary</b>						
Instance ID i-06a3f451908156cc0 (Jenkins_Server)			Public IPv4 address 44.204.38.31   <a href="#">open address</a>			

**Instances (1/2)** [Info](#)

Search

Instance state = running X Clear filters

	Name	Instance ID	Instance state	Instance type	Statu
<input type="checkbox"/>	Jenkins_Server	i-06a3f451908156cc0	Running	t2.micro	2/
<input checked="" type="checkbox"/>	Testing_Server	i-095c6b04cca499258	Running	t2.micro	2/

#### Instance: i-095c6b04cca499258 (Testing\_Server)

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
<div>Instance summary <a href="#">Info</a></div> <div> <div>Instance ID</div> <div>i-095c6b04cca499258 (Testing_Server)</div> </div> <div> <div>Public IPv4 address</div> <div>34.195.199.207   <a href="#">open address</a></div> </div> <div> <div>IPv6 address</div> <div></div> </div> <div> <div>Instance state</div> <div></div> </div>						

- Login to Jenkins\_server machine and install pre-requisites for Jenkins

```
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
```

I've launched an ec2 instance of type amazon linux so installed jdk from amazon-linux-extras

1. amazon-linux-extras list
2. amazon-linux-extras install java-openjdk11 -y
3. yum install jenkins -y
4. service jenkins status
5. service jenkins start
6. service jenkins status
7. netstat -tuln
8. chkconfig jenkins on
9. cat /var/lib/jenkins/secrets/initialAdminPassword

Access Jenkins GUI by allowing port 8080 in the security group (Source I've given as 0.0.0.0/0 for demonstration purpose)

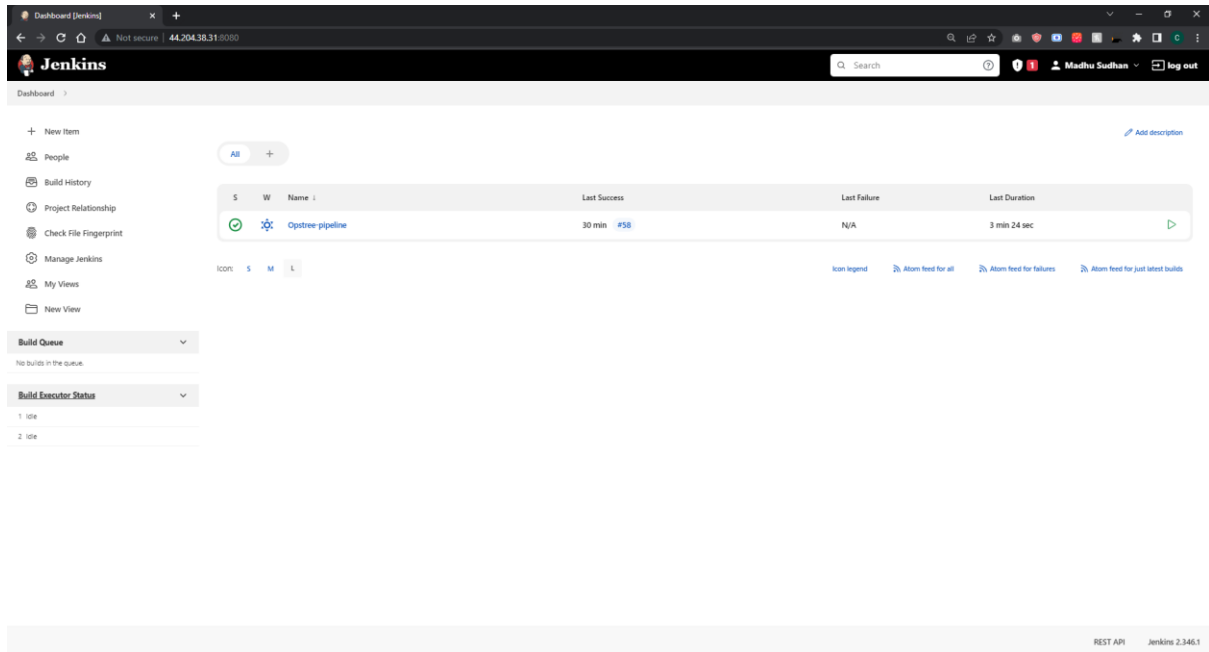
#### Instance: i-06a3f451908156cc0 (Jenkins\_Server)

<div>Security groups</div> <div>sg-0f7e765f057408d7a (launch-wizard-5)</div>																			
<div>Inbound rules</div> <div> <div>Filter rules</div> <table> <tr> <th>Security group rule ID</th><th>Port range</th><th>Protocol</th><th>Source</th><th>Security groups</th></tr> <tr> <td>sgr-062ed93db388868d5</td><td>22</td><td>TCP</td><td>0.0.0.0/0</td><td>launch-wizard-5</td></tr> <tr> <td>sgr-0d994e248ad6e1940</td><td>8080</td><td>TCP</td><td>0.0.0.0/0</td><td>launch-wizard-5</td></tr> </table> </div>					Security group rule ID	Port range	Protocol	Source	Security groups	sgr-062ed93db388868d5	22	TCP	0.0.0.0/0	launch-wizard-5	sgr-0d994e248ad6e1940	8080	TCP	0.0.0.0/0	launch-wizard-5
Security group rule ID	Port range	Protocol	Source	Security groups															
sgr-062ed93db388868d5	22	TCP	0.0.0.0/0	launch-wizard-5															
sgr-0d994e248ad6e1940	8080	TCP	0.0.0.0/0	launch-wizard-5															

Install the plugins suggested by Jenkins when the setup is done for the first type

Create a user name and password and then log in

Created a Pipeline project called - [Opstree-pipeline](#)



For two inputs – Instance\_Id and Instance\_Type, I've given them as parameterized inputs in Jenkins

The screenshot shows the Jenkins configuration page for a 'String Parameter'. The page has a checkbox labeled 'This project is parameterized' which is checked. There are two parameter configuration blocks. The first block is for 'Instance\_Id' with a default value of 'f095c6b04cca499258'. The second block is for 'Instance\_Type' with a default value of 't2.nano'. Each block has a 'Name' field, a 'Default Value' field, and a 'Description' field. There is also a 'Trim the string' checkbox for each parameter.

As this is a pipeline project, We need git to pull the code from the remote repository

For credentials – I've configured username and password as global credentials

## Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	Github_creds	madhusudhan7492/***** (Github credentials)

### Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/madhusudhan7492/Opstree-task.git

Credentials ?

madhusudhan7492/\*\*\*\*\* (Github credentials)

+ Add

Advanced...

Add Repository

Branches to build ?

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/master

Add Branch

Repository browser ?

(Auto)

Additional Behaviours





Add

Script Path ?

Jenkinsfile

As Jenkins has to run aws CLI commands so configured aws-keys in global credentials







## Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	Github_creds	madhusudhan7492/***** (Github credentials)
		Jenkins	(global)	aws-key	AKIASOZVC ..... /***** (aws_access_key_id and password)

Below is the Jenkins file: <https://github.com/madhusudhan7492/Opstree-task/blob/master/Jenkinsfile>

1. Checkout SCM – clone the given repo in Jenkins workspace and use Github\_creds which were configured as global credentials
2. Describe the instance and stop it – how many instances are there in the account and then stop the instance\_id which was given as input
3. Change the instance\_type which was given as input
4. Start the instance
5. Run the ansible-playbook (for ansible I've created ansadmin user in jenkins\_server and testing\_server then created and copied ssh public key from ansadmin user in jenkins\_server and pasted it in testing\_server)
  - a. In jenkins\_server
  - b. Install ansible from amazon-linux-extras in Jenkins\_server
    - i. useradd ansadmin
    - ii. passwd ansadmin
    - iii. su – ansadmin
    - iv. ssh-keygen -t rsa
    - v. Copied id\_rsa.pub (public key) and paste in target node (testing\_server)
  - c. In testing\_server
    - i. useradd ansadmin
    - ii. passwd ansadmin
    - iii. su – ansadmin
    - iv. mkdir .ssh
    - v. cd .ssh
    - vi. vi authorized\_keys
    - vii. Paste the copied public key
    - viii. Change the permission of .ssh folder and authorized\_keys file
    - ix. chmod 700 .ssh (read, write and execute permission for ansadmin)
    - x. chmod 600 authorized\_keys (read and write permission for ansadmin)
  - d. In jenkins\_server (control node) and testing\_server (target node) to check the connectivity
    - i. ssh ansadmin@TargetnodeIP
  - e. In jenkins\_server and testing\_server we need to give sudo root access to ansadmin so adding to wheel group
  - f. Visudo from the root and uncomment - %wheel ALL=(ALL) NOPASSWD: ALL
    - i. usermod -a -G wheel ansadmin (adding secondary group as the wheel)
  - g. Also given ansadmin username and password as global credentials
  - h. In ansible playbook, I've written a task to install httpd and start the service in main.yaml in the dev host group (I've given testing\_server IP in /etc/ansible/hosts file)

## Credentials

T	P	Store i	Domain	ID	Name
		Jenkins	(global)	Github_creds	madhusudhan7492/***** (Github credentials)
		Jenkins	(global)	aws-key	AKIASOZVOLNMTT7PODVH/***** (aws_access_key_id and password)
		Jenkins	(global)	ansadmin	ansadmin

- Once the ansible-playbook has run, next step is to stop the instance and flip to the previous instance type (here I've considered an env variable as OLD\_INSTANCE\_TYPE as "t2.micro")
- For describe-instances, start instance, stop instance, modify instance type – I've configured AWS Access key and AWS Secret key as global credentials and imported @Library('github.com/releaseworks/jenkinslib') \_ - This will load the Releaseworks Jenkins library, that includes the helper function that we will use. And then used withCredentials to get the aws-key and executed commands on specified region and also to execute from jenkins user installed docker and added jenkins user to docker.sock and changed permission to docker.sock file - chmod 777 /var/run/docker.sock
- As part of the plugins I've installed Ansible plugin and given the location of ansible /usr/bin/ in the Global Tool configuration
- Link to latest build output - <http://44.204.38.31:8080/job/Opstree-pipeline/58/consoleText>

```
@Library('github.com/releaseworks/jenkinslib') _
pipeline {
    agent any

    environment {
        OLD_INSTANCE_TYPE = 't2.micro'
    }

    stages {
        // Clone the github repo on jenkins workspace
        stage('Checkout SCM') {
            steps {
                checkout([
                    $class: 'GitSCM', branches: [
                        [name: '**/master']
                    ], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [
                        [credentialsId: 'Github_creds', url: 'https://github.com/madhusudhan7492/Opstree-task.git']
                    ]
                ])
            }
        }

        //Describe the instance status and stop the instance
        stage("Describe and stop the instance") {
            steps {
                withCredentials([
                    [
                        $class: 'UsernamePasswordMultiBinding', credentialsId: 'aws-key', usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY'
                    ]
                ]) {
                    AWS("--region=us-east-1 ec2 describe-instances --query 'Reservations[*].Instances[*].{PublicIP:PublicIpAddress,Name:Tags[?Key==\"Name\"]}[0].Value,Status:State.Name,InstanceId}'")
                    AWS("--region=us-east-1 ec2 stop-instances --instance-ids $InstanceId")
                }
                sh "echo 'Wait for 25 seconds to let the instance turn into stop state'"
                sh "sleep 25"
            }
        }

        //change the instance type as per the input parameter given before running the job
        stage("Change the instance type") {
            steps {
                withCredentials([
                    [
                        $class: 'UsernamePasswordMultiBinding', credentialsId: 'aws-key', usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY'
                    ]
                ]) {
                    AWS("--region=us-east-1 ec2 describe-instances --instance-ids $InstanceId --query 'Reservations[*].Instances[*].{PublicIP: PublicIpAddress, Name:Tags[?Key== 'Name']}][0]')
                    AWS("--region=us-east-1 ec2 modify-instance-attribute --instance-id $InstanceId --instance-type '{\"Value\": \"${Instance_Type}\"}'")
                    AWS("--region=us-east-1 ec2 describe-instances --instance-ids $InstanceId --query 'Reservations[*].Instances[*].{PublicIP: PublicIpAddress, Name:Tags[?Key== 'Name']}][0]')
                }
            }
        }

        //Start the instance (testing_server)
        stage("Start the instance") {
            steps {
                withCredentials([
                    [
                        $class: 'UsernamePasswordMultiBinding', credentialsId: 'aws-key', usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY'
                    ]
                ]) {
                    AWS("--region=us-east-1 ec2 start-instances --instance-ids $InstanceId")
                    sh "sleep 25"
                    AWS("--region=us-east-1 ec2 describe-instances --instance-ids $InstanceId --query 'Reservations[*].Instances[*].{PublicIP: PublicIpAddress, Name:Tags[?Key== 'Name']}][0]')
                }
            }
        }

        //Run ansible playbook on the testing_server
        stage("run ansible playbook") {
            steps {
                //here main.yml file is in the cloned repository
                ansiblePlaybook credentialsId: 'ansadmin', disableHostKeyChecking: true, installation: 'ansible', playbook: 'main.yml'
            }
        }
    }
}
```

```

stage("Stop the instance and flip back to previous instance type") {
  steps {
    withCredentials([
      [class: 'UsernamePasswordMultiBinding', credentialsId: 'aws-key', usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY']
    ]) {
      AWS("--region=us-east-1 ec2 stop-instances --instance-ids $Instance_Id")
      sh "sleep 25"
      AWS("--region=us-east-1 ec2 describe-instances --instance-ids $Instance_Id --query 'Reservations[*].Instances[*].PublicIP: PublicIpAddress, Name: Tags[?Key== 'Name'][][0]')
      AWS("--region=us-east-1 ec2 modify-instance-attribute --instance-id $Instance_Id --instance-type '({\"Value\": \"${OLD_INSTANCE_TYPE}\")')")
      AWS("--region=us-east-1 ec2 start-instances --instance-ids $Instance_Id")
      sh "sleep 25"
      AWS("--region=us-east-1 ec2 describe-instances --instance-ids $Instance_Id --query 'Reservations[*].Instances[*].PublicIP: PublicIpAddress, Name: Tags[?Key== 'Name'][][0]')
    }
  }
}

```

## Demo:

1. Build with the below parameters

**Jenkins**

Dashboard > Opstree-pipeline >

**Pipeline Opstree-pipeline**

This build requires parameters:

Instance\_Id:

Instance\_Type:

**Build**

**Build History** trend

Filter builds...

Stage	Declarative: Checkout SCM	Checkout SCM	Describe and stop the instance	Change the instance type	Start the instance	run ansible playbook	Stop the instance and flip back to previous instance type
Average stage times: (Average full run time: ~3min 33s)	293ms	191ms	44s	23s	41s	7s	1min 32s
#59 Jun 29 02:14 2 commits	293ms	191ms	44s	23s	41s	7s	1min 32s



BUILD\_OUTPUT.txt

## Build output:

## Further improvements:

1. Instead of one playbook, writing ansible roles

2. Instead of one instance id, writing for loop and using aws describe\_instances method to get all the instance\_ids for that region and change the instance\_type and invoke then Ansible role to them
3. We can also take region as an input parameter and invoke that in Jenkinsfile
4. In Jenkinsfile I've hardcoded the OLD\_INSTANCE\_TYPE, which can be further improved by running aws describe\_instances and getting the instance type and storing it in some variable, and then calling it while flipping back